mathematics

MDPI

Article

Novel Reinforcement Learning Research Platform for Role-Playing Games

Petra Csereoka, Bogdan-Ionuţ Roman, Mihai Victor Micea and Călin-Adrian Popa *

Department of Computer and Software Engineering, Polytechnic University Timişoara, Blvd. V. Pârvan, No. 2, 300223 Timişoara, Romania
* Correspondence: calin.popa@cs.upt.ro

Abstract: The latest achievements in the field of reinforcement learning have encouraged the development of vision-based learning methods that compete with human-provided results obtained on various games and training environments. Convolutional neural networks together with Q-learning-based approaches have managed to solve and outperform human players in environments such as Atari 2600, Doom or StarCraft II, but the niche of 3D realistic games with a high degree of freedom of movement and rich graphics remains unexplored, despite having the highest resemblance to real-world situations. In this paper, we propose a novel testbed to push the limits of deep learning methods, namely an OpenAI Gym-like environment based on Dark Souls III, a notoriously difficult role-playing game, where even human players have reportedly struggled. We explore two types of architectures, Deep Q-Network and Deep Recurrent Q-Network, providing the results of a first incursion into this new problem class. The source code for the training environment and baselines is made available.

Keywords: Deep Q-Network; Deep Recurrent Q-Network; Dark Souls III; video games; visual-based reinforcement learning; neural networks

MSC: 68T07; 68T40

Citation: Csereoka, P.; Roman, B.-I.; Micea, M.V.; Popa, C.-A. Novel Reinforcement Learning Research Platform for Role-Playing Games. *Mathematics* **2022**, *10*, 4363. https://doi.org/10.3390/math10224363

Academic Editors: Adrian Sergiu Darabant and Diana-Laura Borza

Received: 12 October 2022
Accepted: 17 November 2022
Published: 20 November 2022

1. Introduction

Learning can take various forms, but the primary source of information still remains our interaction with the environment. A significant fraction of the inputs we receive comes in the form of visual signals that are processed and decoded and later reasoned upon. With the latest developments in processing power, computers are now able to process and refine more raw data than before [1], yet humans still have the major advantage of better retaining and organizing the information they accumulate over time, combining it in novel ways to solve problems that arise dynamically [2]. This difference between humans and computer models is even more pronounced when it comes to developing strategies in complex environments that are changing fast and are rich in details. Neural networks have been applied successfully to a wide range of tasks [3,4], making them a potentially strong contender for solving current real-world problems.

Several models have been proposed over time to address this gap; however, the first success story in visual learning is represented by Deep Q-Network (DQN) architectures [5,6] that have managed to achieve human-level performance in several Atari 2600 games. This was made possible by framing the problem as a Markov decision process (MDP) [7] and employing convolutional neural networks (CNNs) to build a control policy using a Q-Learning variant [8]. The neural network thus obtained has managed to learn the mappings between input states and relevant actions and their corresponding values with the help of stochastic gradient descent, experience replay mechanisms and exploration and exploitation techniques.

This breakthrough marked an important milestone, as previous methods assumed complete knowledge about the environment, its rules and states and were fine-tuned for particular scenarios, but the DQN solution was a step towards tackling more generic problems closer to real-world environments. Encouraged by this development, new variants of DQNs appeared, and new test environments were proposed to further push the limits of Q-Learning. An important limitation is related to the problem of observability. Most real world problems involve partial knowledge about the global state of the environment, and, in most cases, the observations that can be recorded are available for a limited time only. Hence, reasoning upon longer sequences of events becomes harder, with detecting recipes that span over longer periods of time being very costly if not impossible.

To mitigate such shortcomings, a new DQN variant was proposed: Deep Recurrent Q-Network (DRQN) [9]. The changes made to the original DQN model were minimal and localized, and the convolutional layers remained unchanged; however, the first fully-connected layer was replaced with a recurrent one containing LSTM cells. This new addition yielded a model able to better integrate information over time, thus making it a more suitable contender for problems that involve longer sequences of actions and a better generalization power for partially observable scenarios. This increase in performance, on the other hand, came with more training time and higher hardware requirements.

Dark Souls III [10] is one of the most notoriously difficult role-playing games even for human players, with statistics collected across the entire user base showing only 83.9% of the players that acquire the game manage to pass the entry zone and only 3.6% finish the game with all the achievements [11]. Their experiences with the challenges vary depending on past encounters with similar games, but nonetheless, all players must familiarize themselves with different warrior classes and enemies with complex movements. The game combines difficult boss battles, layered environments with many hidden secrets and diverse combat strategies that vary with each new enemy encountered.

Our work comes to address the research gap between existing training and testing environments created for deep reinforcement learning. Previously implemented gyms were based on simple games, often with 2D imagery, lacking the challenges agents must face in the real world, which is rich in details and offers a high degree of movement in a refined manner. To the best of our knowledge, this paper provides the first environment created having in mind the main limitations of existing training environments in terms of graphics and degrees of movement for the agents, attempting to boost the work done in problems approaching the real-world level in visual deep learning.

In the next sections, we present our initial results obtained by training an agent to defeat the first major boss, Iudex Gundyr, using DQNs and DRQNs in an OpenAI-type gym. Being a closed-source game, extracting all the necessary information for creating the mission as well as assigning proper rewards requires the implementation of helper programs and scripts to access and modify the data stored in RAM and map the location of each relevant entry. Our results are promising, with the time required to defeat the enemy being comparable to the times obtained by players new to this genre.

This paper makes the following contributions:

- An analysis of existing training and testing environments for deep reinforcement learning in problems that can be modeled with the help of games in a pursuit of reaching lifelong learning;
- A complete training and testing environment, including all the Python source files for the gym, automated RAM address retrieval in the form of Cheat Engine scripts, .dll files needed for a complete interaction between the agent and the game, and documentation on how to use different components and their relationship;
- The implementation of state-of-the-art baselines that match the performance of human newcomers to the game genre, showing that value-based methods can be deployed successfully even for more visually complex problems, and opening the path at the same time for further research using different deep learning approaches.

In the following section, we discuss the state-of-the-art training and testing environments available for researchers pursuing to create agents for reinforcement learning problems. Then, selected baselines are detailed, together with the main components and functional details of the novel proposed gym and mission. These are followed by the initial experimental results obtained for the first mission that players face in normal game mode, and future improvements and conclusions round the analysis. The source code for the training environment and baselines can be found at: https://github.com/Bo-roman/DSIII-RL (accessed on 12 October 2022).

2. Related Work

DQNs have been first applied successfully to Atari 2600 games [5,6] and later similar methods were implemented to solve more complex tasks with good results. To improve on the convergence during training, a series of methods have been applied, such as experience replay mechanisms [12], separating the target network to reduce overestimation of the Q-values, reducing the action space and compressing the input by clipping and conversion to grayscale. However, a large number of the games comprised in this environment ensure total state observability and can be completed within short timespans, rendering it a good starting benchmark [13].

Later, a more realistic training environment was proposed, based on the first-person shooter video game called Doom. Unlike Atari 2600, this platform provided a first-person perspective and more realistic 3D imagery with customizable mechanisms, offering a higher degree of flexibility than previous environments [14]. The agents still had to make decisions based only on the visual information provided by the game, but a series of tasks with various difficulties were proposed over the years, further proving the versatility of DQN approaches: Basic scenarios [14,15], with simple primitive movements and enemy confrontation, Maze navigation [16,17], which requires escaping from custom made complex mazes, Defend the center [18], in which the agent is spawned in the middle of a room and must defend only by performing rotations, DeathMatch scenarios [19–22], which prompted agents to kill as many opponents as possible within a fixed time frame, etc.

Another challenging environment proposed during the same period is represented by the StarCraft family of games [23], providing denser state spaces and a stronger focus on multi-agent tasks. By organizing a high number of bot competitions, the environment gained more popularity, and hence encouraged a constant improvement of the proposed models each year [24]. The full game requires players to gather resources, construct buildings and create an army, and finally eliminate the opponents. The decisions taken at each step influence long term outcomes and the duration of a game can vary. Initial solutions focused on solving either the micro level, namely multi-unit collaboration problems [25], or the macro level, tackling decision-making issues within big state spaces [26]. Later, hierarchical solutions emerged combining the targets previously separated and solved in isolation [27,28].

In recent years, with the creation of the MALMO platform [29], a new horizon has been opened in terms of training and testing environments. Built on top of the open world game called Minecraft, a series of OpenAI Gym-type environments [30] emerged focusing on a variety of tasks, which were not analyzed before, allowing researchers to build agents for long horizon missions, collaborative, and creative problems. The game offers a high degree of freedom in creating missions in an environment which follows the laws of the real world, and with crowd sourced datasets the range of approaches that can be implemented was widened [31,32].

At the same time, a new research direction in the last two years was focused around the topic of coordination in multi-agent scenarios, modeled and experimented with as an instance of the popular card game Hanabi [33]. Games have proved to be important testing and training environments for studying how complex decision making problems can be attempted, and Hanabi forces players to plan ahead, reason at higher levels about beliefs and tactics to achieve a common goal, under time and communication restrictions. The

open-source learning environment has gained popularity fast and is now considered a benchmark for machine learning coordination solutions [34,35].

While existing environments enable researchers to test and implement a wide variety of methods for a significant number of missions, starting from simple tasks to scenarios suitable for lifelong learning investigations, they still are strongly anchored in the virtual, and hence cannot model the continuous nature of visual transitions and phenomena from the real world, due to the graphics and movement rules of different entities. The new environment that is proposed comes to address these issues with its 3D immersive graphics and continuous movements, additionally providing starting baselines that can he enhanced in the future.

3. Proposed Methodology

3.1. Mission Description

The first important confrontation within the game is represented by the Cemetery of Ash mission, where the players are facing the entity called Iudex Gundyr. At the start, this enemy has a tall humanoid form with heavy armor and a long halberd; however, after his health is lowered below a certain threshold, the entity will transform into a black monster that is significantly bigger and has a new set of attacks. Initially, the players are confronted with the unique fighting style of the enemy, using a wide range of attacks and combinations that must be dodged or sidestepped in order to survive. During the second phase, the speed and range of the entity are further increased, making it even more difficult to defeat.

Players can choose from 10 warrior classes with various stats, such as endurance, vitality, strength, dexterity, intelligence, faith, defense, resistance, etc., having a great impact on what type of actions the players can take in the first parts of the game. As a starting class, most players recommend either selecting a ranged type, allowing players to attack from farther away and thus avoid the damage caused by most attacks, or the Knight class, as it comes with the most durable armor and shield, as well as a melee weapon in the form of a sword. For our experiments, we have opted for the Knight class, but, to raise the difficulty of the task, we have removed from the list of available actions the parry commands (see Figure 1).

Figure 1. The Knight class warrior battling with Iudex Gundyr in the first phase (**left**) and in the second phase (**right**) [10].

Under normal circumstances, the camera can be pointed in any direction with the help of the mouse, but, to allow the agent to better focus on the mission, we have opted for a Lock Target mechanism which will keep the camera pointed towards the enemy, thus freeing the agent from having to manage that as well.

3.2. Retrieving Attributes for Training

As a first step, to be able to train agents, the relevant parameters from the game must be identified. There are two entities involved: the player or agent and the enemy. In a normal game mode, these classes have a larger set of attributes that are read and updated repeatedly; however, for training purposes, we have opted to reduce these to the bare

minimum set that enables us to reinitialize the scenario, grant appropriate rewards, and provide the relevant metrics to the agent.

In order to build the scenario, we need access to:

- The player's health. To start the mission, the health attribute of the player must be reduced to zero to trigger the initialization cycle, which teleports the player to the last checkpoint and resets the map and the attributes of the entities that are present, including the player. This operation is usually performed as part of the environment reset function from the API, and this is the fastest way to achieve it from an operational point of view;
- The player's position along the three axes (X, Y, Z). The last checkpoint is farther than our targeted enemy; hence, by manipulating the coordinates, we can ensure that the agent is moved closer, within the range of the entity. This operation reduces the time wasted between fights, allowing a better use of the available computational resources, and the agent can focus on the actual combat part as opposed to wandering connecting paths with no added bonus;
- The enemy's state: defeated or encountered. In order to trigger the start of the confrontation, the enemy's state must be set to encountered, however, after the initialization cycle, this attribute has a different value, and the player would have to go through numerous interactions to alter it. Therefore, to reduce the time needed to set up the mission, we overwrite this attribute automatically to the desired value.

For awarding the rewards, we have decided to take into account the health points of the player and the enemy, and the player's endurance. For every successful attack, the health of the enemy will be reduced, hence resulting in positive rewards, with the maximum amount when the enemy is killed; if the agent does not retreat to a safe area from attacks, their health attributes are reduced, resulting in negative rewards and ultimately their death scoring the most severe penalty.

These attributes are not directly accessible, as the game does not have an API for parameter and entity manipulations such as existing gyms; hence, they must be extracted and overwritten directly in the RAM memory. Exploring the RAM memory for relevant information is hindered by the protection mechanisms built within the operating system such as Address Space Layout Randomization [36], which makes previously identified memory locations no longer useful at the next training session when we restart the game or the gym. To overcome that, static offsets have to be identified each time, and this can be achieved automatically with the help of the open-source application called Cheat Engine [37].

Cheat Engine provides basic functionalities, such as scanning and modifying the memory zones of running processes within the system, but it also provides more advanced features, such as creating scripts with the help of the integrated Lua interpreter to automate many of the tasks that would otherwise have to be done manually at each run. Previously identified memory zones, offsets, and scripts are stored in a cheat table, and, to identify new locations, methods such as a value scan and a pointer scan can be deployed. The memory locations for certain attributes are easy to find, as their values are shown in the graphical user interface of the game; however, other parameters require extensive search and even interactions with the keys to alter the values and hence aid the search.

3.3. Gym Creation

To build our training environment, we have opted for a standardized formatting; namely, we have extended the `Env` interface from the OpenAI Gym libraries. Most of the methods within the class require a form of interaction with the environment, but, given the nature of the underlying game, in our case, this is only possible via intermediaries, such as Cheat Engine, which functions to retrieve attributes and alter the state of the game, and a community-provided .dll file to allow us to emulate physical key presses from the keyboard with software instructions. The communication between these components is

realized with the help of sockets, with the data being structured into JSON bursts in a custom-built formatting.

The list of selected actions for a mission is provided via a configuration file, but definitions within the gym allow users to select from various types of movement, rolls, attacks, and interactions with objects. Within the same configuration file, hyperparameter values, limits, and sizes can be specified, allowing researchers to have full control over the problem and be able to adapt it to their needs (see Figure 2).

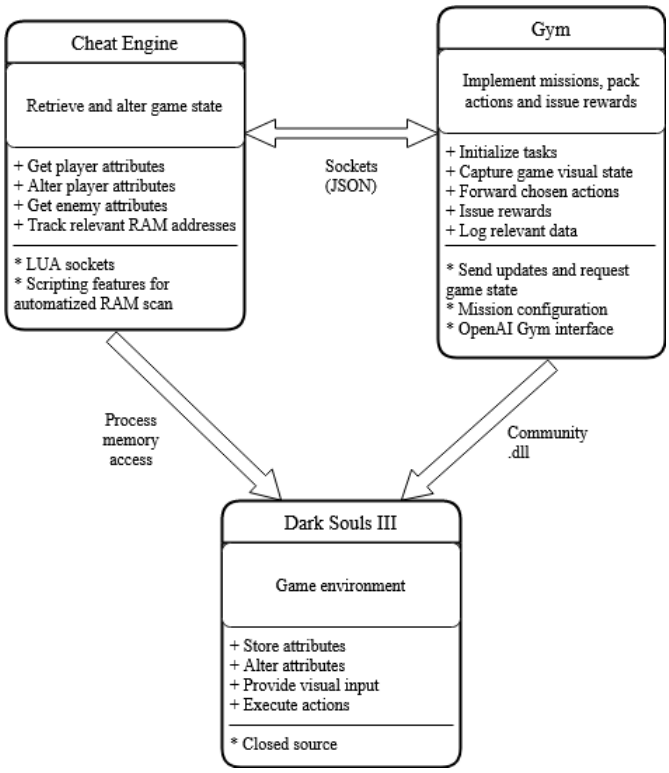

Figure 2. System architecture highlighting main software components for the training and testing environment, as well as main functions and attributes.

3.4. Methods Considered for the Analysis

Given the success of DQN-based architectures in previous training environments, we have considered 2 main approaches for this analysis, namely vanilla and recurrent DQN models. The vanilla network architecture is based on the one proposed by Mnih et al. [5,6] for solving Atari 2600 games, while the recurrent one is closer to the model proposed by Hausknecht et al. [9] (see Figure 3).

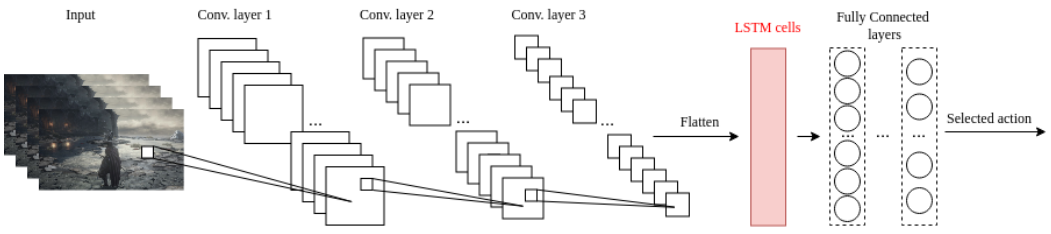

Figure 3. Architectural overview of the models implemented as baselines. The black components are part of both models, while the red LSTM cells are included only in the DRQN variant.

The input images are captured and rescaled to a smaller size while still keeping the main elements visible and distinguishable. Due to the color palette of the game, we have opted to keep the RGB formatting, as after the conversion to grayscale, certain important game elements blended into the background, making it hard to detect them on time.

The feature map extraction was done by 3 consecutive convolutional layers, followed by fully-connected layers to compute the approximation for the Q-values.

To reduce computational costs, gather more system state related information, and build better policies, we have implemented a frame-skipping mechanism with a factor of 4 [38]. To balance exploration and exploitation, we have opted for an ϵ-greedy approach that will prompt the agent to select a random next step with a probability of ϵ and require a deliberate choice otherwise. The starting value was set to 1 and was then annealed to 0.1 to encourage long-term exploration as well.

To ensure convergence, an experience replay memory buffer was used that stores entries as tuples in an SQL Light database. We have opted for this solution as it enables us to store and conveniently load samples of experiences at runtime faster; it offers better control of the training process in case of interruptions.

4. Experiments and Results

As described in previous sections, we have selected the first boss battle from the game that begins in the area called Cemetery of Ash. The player enters the region through a narrow passageway and is confronted by an unmoving entity sitting in the middle of a stone circle. The area is surrounded by impassable large stone remnants of walls and hallways, trees with dense overhanging roots, and parts of an ancient cemetery, except for the western region, which is an open edge over a cliff (see Figure 4).

Figure 4. Mission main battle region.

In order to win, the agent has to defeat the enemy; however, they are at disadvantage, as the entity has much higher health, speed, agility, and attack strength and range, while the agent can also fall over the cliff either by making a wrong choice or getting hit by the entity.

One of the main limiting factors for the player's strategy is represented by the endurance attribute. This stat has a direct impact on the amount of stamina the player has to execute actions, such as dodging and swinging weapons, and it also influences damage resistance. During initial experiments, due to exhausting the endurance attribute in early phases of the mission, the agent overfitted to running away from the entity as opposed to rationing the stamina better by alternating attacks with recharging. Hence, to reduce the difficulty of the task, we have set the endurance to an infinite amount, so the agent can focus more on the combat aspect of the mission.

The rewards and penalties we have considered for this mission can be classified into 2 main categories: primary—they represent the main goal of the agent: every successful attack and the defeat of the enemy will yield a positive reward, while every damage accumulated by the agent, including their death, will be punished with negative rewards, and auxiliary—introduced during training to correct unwanted behaviors: if the agent performs a certain number of consecutive movement/attack actions above a threshold, a small penalty is issued to motivate the agent to perform more attacks/dodge and retreat movements. Defeating the entity in its phase 1 form is an important milestone, as when it morphs into its next evolutionary step, the behavior is changed, and the agent has to learn new tactics.

The models receive 4 input images that were scaled down to 150×93 frames while still keeping the RGB formatting. They are then processed by the 3 consecutive convolutional

layers, followed by a variable number of fully connected layers for the DQN architecture and LSTM cells, followed by fully connected layers for the DRQN model.

For the DRQN model, the initial experiments allowed the agent to select from the complete set of actions, and we have implemented a reward-clipping method. The agents trained with this setup have shown a high preference for movement-type actions, resulting in longer episodes with little damage done to the entity and no wins. For the following experiments, we have reduced the action set to movements along the axes, dodges and attacks, increased the training time and changed the reward proportions, obtaining the first win (see Table 1).

Table 1. DRQN model architecture details.

Layer Type	Input Size	Activation	Output Size
Conv1	$150 \times 93 \times 12$	ReLU	$36 \times 22 \times 32$
Conv2	$36 \times 22 \times 32$	ReLU	$17 \times 10 \times 64$
Conv3	$17 \times 10 \times 64$	ReLU	$15 \times 8 \times 64$
LSTM	7680	-	1024
Output	1024	ReLU	No. of actions

For the first model type, we have given a higher reward (double) for attack type actions, as the life of the entity is almost 4 times higher than the one of the agent and to prevent an overfitting on evasive moves, such as those the previous versions did (DRQN-1). This approach resulted in fairly fast wins as opposed to later models; however, they also obtained a lower score, as this aggressive choice also resulted in the agent losing more life points while attacking repeatedly.

The second type was a mirror of the previous model, though this time the evasive moves gained more importance, having in mind that every health unit from the agent is more valuable (DRQN-2); this time the agent had learned combos (chains of actions) that human players also perform in similar scenarios, but it still did not manage to avoid certain type of attacks from the enemy that would have required multiple dodges and putting bigger distance between the agent and Iudex.

The most successful model type (DRQN-3) had 10% more reward for attack moves and had obtained the first victory around the 900-episode milestone, averaging at 10 wins every 500 episodes in the last phases of training (episodes 3000–4500). The agent also managed to reach phase 2 with an increasing tendency, on average in 50 out of every 500 episodes (episodes 1500–4500), see Table 2.

Table 2. Reward table for the DRQN models.

Architecture Type	Frames Trained	Reward for Successful Attack	Reward for Taking Damage from Enemy	Absolute Value for Win/Death
DRQN-1	500,000	+200	−100	2000
DRQN-2	500.000	+100	−200	2000
DRQN-3	500.000	+60	−50	2000

All of these models had 512 LSTM type cells and 2 fully connected layers. The training sessions contained 4500 episodes (training unit lasting from the first state to the ending state), adding up to roughly 500,000 frames for each type.

The DQN model, on the other hand, did not provide such good results. Three architecture types were analyzed: two with a single fully connected layer, but with different sizes, and one with four fully connected layers, mimicking the recurrent model. The first two types managed to reach the second phase in later episodes but never managed to defeat the enemy. The last type succeeded in killing the entity once during episode 1650 in a battle that lasted 50 s and reached the second phase on average 3 times every 500 episodes.

These models were trained on the reduced action set with reward clipping, going through approximately 300,000 frames.

The experiments were performed using an NVIDIA GeForce RTX 2080 Ti GPU and an AMD Ryzen 9 5900X processor, thus marking the hardware requirements at the low-to-medium end of the implementation cost spectrum for machine learning applications. The time complexity of the analyzed models is consistent with the models proposed in the original papers [5,6,9], with an additional $\sim$ 5% overhead due to frame preprocessing and experience replay storing. The GPU load percentage was monitored during training, and the average time for an inference was 104.985 milliseconds for the DRQN model and 1.001 milliseconds for the DQN model (see Tables 3 and 4).

Table 3. DQN model architecture details.

Layer Type	Input Size	Activation	Output Size
Conv1	$150 \times 93 \times 12$	ReLU	$36 \times 22 \times 32$
Conv2	$36 \times 22 \times 32$	ReLU	$17 \times 10 \times 64$
Conv3	$17 \times 10 \times 64$	ReLU	$15 \times 8 \times 64$
FC1	7680	ReLU	128
FC2	128	ReLU	64
FC3	64	ReLU	32
Output	32	ReLU	No. of actions

Table 4. Highest reward and fastest win for the most successful models.

Architecture	Time until win (s)	Total reward	Episode reached
DRQN-1	31.8	**1500**	1991
	24.9	1450	1867
DRQN-2	37.6	**1670**	1130
	24.3	1450	3031
DRQN-3	63	**1900**	3436
	45	1590	4177
DQN	48.7	1650	2147

With only the game running and the agent in an idle state, the GPU utilization revolves around 6%, with a minimal memory utilization around 3%. Once the episode is reset, both the game and the gym must make several steps to prepare the next session; hence, the utilization will rise above 20%, and the memory is also pushed around 13%. During the inference period (during decision making within the episode), the parameters move to around 37% for the GPU load and 25% for the memory. Given these measurements, we can also conclude that less performant GPUs, such as GTX 1050 Ti, are still able to fulfill the minimal requirements for running our proposed gym and the baseline models (see Figure 5).

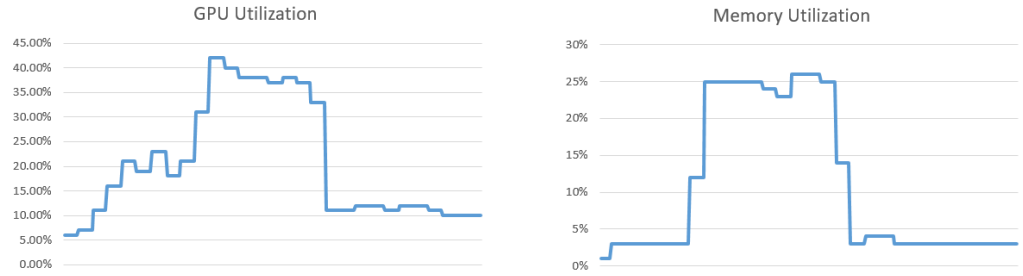

Figure 5. GPU utilization parameters during an episode.

5. Conclusions and Future Work

The currently available training and testing environments for visual learning provide the means for research in specific problem classes, but the niche of more immersive and realistic environments was not yet explored, although they come closer to the way humans perceive visual stimuli. Another important aspect is related to the movement of the agent: in most of the existing environments, actions are performed relative to an existing square grid, which is either hidden by the graphics and existing in the background or revealed, such as in Minecraft, but in the real world, our perception has a more continuous and free nature. The training environment proposed in this paper addresses these two main points, providing the means for a new direction for future research in the field.

The two baselines we have implemented show promising results. During training, both agent types have managed to learn useful combinations of primitive actions that resulted in successful attacks at optimal locations and to avoid many of the long range attacks either by dodging, hiding behind the enemy, or retreating right after a number of successful attacks, thus exploiting weaknesses within the enemy mechanics.

In this paper, we have presented the results obtained by a first incursion into the immersive world of graphically intensive games; however, there are certain aspects to the gym design and models that can be improved in the future.

The player class which we have selected for our experiments has certain advantages and disadvantages compared to the other available options; however, by restricting the list of actions to the basic movements and attacks that all the classes possess, our approach can be easily extended to any of the remaining player types without problems. A more in-depth analysis should be further conducted into the impact of different agent classes on the speed of learning and what attributes impact the success rate the most.

Furthermore, in our current implementation, we have used vanilla ϵ-greedy, but in recent years improved versions for this approach were presented [39], as well as more complex methods to address the problem of exploitation vs. exploration [40]. Similarly, for sampling among gathered experiences, a number of different methods have emerged [41,42] that could aid the training process when the number of available actions is large and the rewards are sparse.

Finally, to automatically point the camera towards the enemy entity, we have employed a target lock mechanism, but this creates certain problems, as Iudex Gundyr in the second phase morphs into a very large entity that is no longer able to fit into the field of view of the agent, which, hence, cannot detect certain attack types in time. Additionally, when the agent is facing the enemy with their back towards the edge of the cliff, choosing an action of the retreat type will inevitably result in the agent dying from falling into the void, as they cannot always see the edge. A better method should be implemented to automatically move the camera to regions of interest.

Author Contributions: Conceptualization: C.-A.P. and M.V.M.; methodology: P.C. and C.-A.P.; software: B.-I.R. and P.C.; validation: P.C., C.-A.P. and M.V.M.; formal analysis: C.-A.P.; investigation: P.C.; resources: C.-A.P. and M.V.M.; data curation: B.-I.R. and P.C.; writing—original draft preparation: P.C.; writing—review and editing: C.-A.P. and M.V.M.; visualization: C.-A.P.; supervision: C.-A.P. and M.V.M.; project administration: P.C. All authors have read and agreed to the published version of the manuscript.

Funding: The APC was funded by Polytechnic University Timişoara, Romania.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Zheng, W.; Yin, L. Characterization inference based on joint-optimization of multi-layer semantics and deep fusion matching network. *PeerJ Comput. Sci.* **2022**, *8*, e908. https://doi.org/10.7717/peerj-cs.908.
2. Zheng, W.; Tian, X.; Yang, B.; Liu, S.; Ding, Y.; Tian, J.; Yin, L. A Few Shot Classification Methods Based on Multiscale Relational Networks. *Appl. Sci.* **2022**, *12*, 4059. https://doi.org/10.3390/app12084059.
3. Qin, X.; Liu, Z.; Liu, Y.; Liu, S.; Yang, B.; Yin, L.; Liu, M.; Zheng, W. User OCEAN Personality Model Construction Method Using a BP Neural Network. *Electronics* **2022**, *11*, 3022. https://doi.org/10.3390/electronics11193022.
4. Stai, E.; Kafetzoglou, S.; Tsiropoulou, E.E.; Papavassiliou, S. A holistic approach for personalization, relevance feedback & recommendation in enriched multimedia content. *Multimed. Tools Appl.* **2018**, *77*, 283–326. https://doi.org/10.1007/s11042-016-4209-1.
5. Mnih, V.; Kavukcuoglu, K.; Silver, D.; Graves, A.; Antonoglou, I.; Wierstra, D.; Riedmiller, M. Playing Atari with Deep Reinforcement Learning. In Proceedings of the NIPS Deep Learning Workshop, Lake Tahoe, NV, USA, 9 December 2013.
6. Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A.A.; Veness, J.; Bellemare, M.G.; Graves, A.; Riedmiller, M.; Fidjeland, A.K.; Ostrovski, G.; et al. Human-level control through deep reinforcement learning. *Nature* **2015**, *518*, 529–533. https://doi.org/10.1038/nature14236.
7. Bellman, R. A Markovian Decision Process. *J. Math. Mech.* **1957**, *6*, 679–684.
8. Watkins, C.J.C.H.; Dayan, P. Q-learning. *Mach. Learn.* **1992**, *8*, 279–292. https://doi.org/10.1007/BF00992698.
9. Hausknecht, M.; Stone, P. Deep Recurrent Q-Learning for Partially Observable MDPs. In Proceedings of the AAAI Fall Symposium on Sequential Decision Making for Intelligent Agents (AAAI-SDMIA15), Arlington, TX, USA, 12–14 November 2015.
10. Available online: https://store.steampowered.com/app/374320/DARK_SOULS_III/ (accessed on 15 August 2022).
11. Available online: https://steamcommunity.com/stats/374320/achievements (accessed on 15 August 2022).
12. Fedus, W.; Ramachandran, P.; Agarwal, R.; Bengio, Y.; Larochelle, H.; Rowland, M.; Dabney, W. Revisiting Fundamentals of Experience Replay. In Proceedings of the International Conference on Machine Learning (ICML), Online, 13–18 July 2020.
13. Fan, J. A Review for Deep Reinforcement Learning in Atari:Benchmarks, Challenges, and Solutions. *arXiv* **2021**, arXiv:2112.04145. https://doi.org/10.48550/ARXIV.2112.04145.
14. Kempka, M.; Wydmuch, M.; Runc, G.; Toczek, J.; Jaskowski, W. ViZDoom: A Doom-based AI research platform for visual reinforcement learning. In Proceedings of the 2016 IEEE Conference on Computational Intelligence and Games (CIG), Santorini, Greece, 20–23 September 2016; IEEE: Piscataway, NJ, USA, 2016. https://doi.org/10.1109/CIG.2016.7860433.
15. Adil, K.; Jiang, F.; Liu, S.; Grigorev, A.; Gupta, B.; Rho, S. Training an Agent for FPS Doom Game using Visual Reinforcement Learning and VizDoom. *Int. J. Adv. Comput. Sci. Appl.* **2017**, *8*, 32–41. https://doi.org/10.14569/IJACSA.2017.081205.
16. Kulkarni, T.D.; Saeedi, A.; Gautam, S.; Gershman, S.J. Deep Successor Reinforcement Learning. *arXiv* **2016**, arXiv:1606.02396. https://doi.org/10.48550/ARXIV.1606.02396.
17. Woubie, A.; Kanervisto, A.; Karttunen, J.; Hautamaki, V. Do Autonomous Agents Benefit from Hearing? *arXiv* **2019**, arXiv:1905.04192. https://doi.org/10.48550/ARXIV.1905.04192.
18. Schulze, C.; Schulze, M. ViZDoom: DRQN with Prioritized Experience Replay, Double-Q Learning and Snapshot Ensembling. In Proceedings of the SAI Intelligent Systems Conference, London, UK, 6–7 September 2018; Springer International Publishing: Berlin/Heidelberg, Germany, 2018; pp. 1–17. https://doi.org/10.1007/978-3-030-01054-6_1.
19. Zakharenkov, A.; Makarov, I. Deep Reinforcement Learning with DQN vs. PPO in VizDoom. In Proceedings of the 2021 IEEE 21st International Symposium on Computational Intelligence and Informatics (CINTI), Budapest, Hungary, 18–20 November 2021; IEEE: Piscataway, NJ, USA, 2021. https://doi.org/10.1109/cinti53070.2021.9668479.
20. Lample, G.; Chaplot, D.S. Playing FPS Games with Deep Reinforcement Learning. In Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, San Francisco, CA, USA, 4–9 February 2017.
21. Bhatti, S.; Desmaison, A.; Miksik, O.; Nardelli, N.; Siddharth, N.; Torr, P.H.S. Playing Doom with SLAM-Augmented Deep Reinforcement Learning. *arXiv* **2016**, arXiv:1612.00380. https://doi.org/10.48550/ARXIV.1612.00380.
22. Wydmuch, M.; Kempka, M.; Jaskowski, W. ViZDoom Competitions: Playing Doom From Pixels. *IEEE Trans. Games* **2019**, *11*, 248–259. https://doi.org/10.1109/tg.2018.2877047.
23. Vinyals, O.; Ewalds, T.; Bartunov, S.; Georgiev, P.; Vezhnevets, A.S.; Yeo, M.; Makhzani, A.; Küttler, H.; Agapiou, J.; Schrittwieser, J.; et al. StarCraft II: A New Challenge for Reinforcement Learning. *arXiv* **2017**, arXiv:1708.04782. https://doi.org/10.48550 /ARXIV.1708.04782.
24. Certicky, M.; Churchill, D.; Kim, K.J.; Certicky, M.; Kelly, R. StarCraft AI Competitions, Bots, and Tournament Manager Software. *IEEE Trans. Games* **2019**, *11*, 227–237. https://doi.org/10.1109/tg.2018.2883499.
25. Usunier, N.; Synnaeve, G.; Lin, Z.; Chintala, S. Episodic Exploration for Deep Deterministic Policies: An Application to StarCraft Micromanagement Tasks. In Proceedings of the International Conference on Learning Representations (ICLR), Toulon, France, 24–26 April 2017.
26. Xu, S.; Kuang, H.; Zhuang, Z.; Hu, R.; Liu, Y.; Sun, H. Macro action selection with deep reinforcement learning in StarCraft. In Proceedings of the Fifteenth AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE), Atlanta, GA, USA, 8–12 October 2019.

27. Liu, T.; Wu, X.; Luo, D. A Hierarchical Model for StarCraft II Mini-Game. In Proceedings of the 2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA), Boca Raton, FL, USA, 16–19 December 2019; IEEE: Piscataway, NJ, USA, 2019. https://doi.org/10.1109/icmla.2019.00042.
28. Hu, Y.; Li, J.; Li, X.; Pan, G.; Xu, M. Knowledge-Guided Agent-Tactic-Aware Learning for StarCraft Micromanagement. In Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence (IJCAI), Stockholm, Sweden, 13–19 July 2018; International Joint Conferences on Artificial Intelligence Organization: CA, USA, 2018. https://doi.org/10.24963/ijcai.2018/204.
29. Johnson, M.; Hofmann, K.; Hutton, T.; Bignell, D. The Malmo Platform for Artificial Intelligence Experimentation. In Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence (IJCAI), New York, NY, USA, 9–15 July 2016.
30. Brockman, G.; Cheung, V.; Pettersson, L.; Schneider, J.; Schulman, J.; Tang, J.; Zaremba, W. OpenAI Gym. *arXiv* **2016**, arXiv:1606.01540. https://doi.org/10.48550/ARXIV.1606.01540.
31. Guss, W.H.; Houghton, B.; Topin, N.; Wang, P.; Codel, C.; Veloso, M.; Salakhutdinov, R. MineRL: A Large-Scale Dataset of Minecraft Demonstrations. In Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence (IJCAI), Macao, China, 10–16 August 2019; International Joint Conferences on Artificial Intelligence Organization: 2019. https: //doi.org/10.24963/ijcai.2019/339.
32. Gray, J.; Srinet, K.; Jernite, Y.; Yu, H.; Chen, Z.; Guo, D.; Goyal, S.; Zitnick, C.L.; Szlam, A. CraftAssist: A Framework for Dialogue-enabled Interactive Agents. *arXiv* **2019**, arXiv:1907.08584. https://doi.org/10.48550/ARXIV.1907.08584.
33. Bard, N.; Foerster, J.N.; Chandar, S.; Burch, N.; Lanctot, M.; Song, H.F.; Parisotto, E.; Dumoulin, V.; Moitra, S.; Hughes, E.; et al. The Hanabi challenge: A new frontier for AI research. *Artif. Intell.* **2020**, *280*, 103216. https://doi.org/10.1016/j.artint.2019.103216.
34. Muglich, D.; de Witt, C.S.; van der Pol, E.; Whiteson, S.; Foerster, J. Equivariant Networks for Zero-Shot Coordination. In Proceedings of the Advances in Neural Information Processing Systems (NeurIPS). *arXiv* **2022**, arXiv.2210.12124. https: //doi.org/10.48550/arXiv.2210.12124.
35. Grooten, B.; Wemmenhove, J.; Poot, M.; Portegies, J. Is Vanilla Policy Gradient Overlooked? Analyzing Deep Reinforcement Learning for Hanabi. In Proceedings of the AAMAS Adaptive and Learning Agents Workshop. *arXiv* **2022**, arXiv:2203.11656. https://doi.org/10.48550/ARXIV.2203.11656.
36. Jia, X.; Bin, Z.; Chao, F.; Chaojing, T. An Automatic Evaluation Approach for Binary Software Vulnerabilities with Address Space Layout Randomization Enabled. In Proceedings of the 2021 International Conference on Big Data Analysis and Computer Science (BDACS), Kunming, China, 25–27 June 2021; IEEE: Piscataway, NJ, USA, 2021. https://doi.org/10.1109/bdacs53596.2021.00045.
37. Developers, C.E. Cheat Engine. Available online: https://www.cheatengine.org/ (accessed on 15 August 2022).
38. Kalyanakrishnan, S.; Aravindan, S.; Bagdawat, V.; Bhatt, V.; Goka, H.; Gupta, A.; Krishna, K.; Piratla, V. An Analysis of Frame-skipping in Reinforcement Learning. *arXiv* **2021**, arXiv:2102.03718. https://doi.org/10.48550/ARXIV.2102.03718.
39. Dabney, W.; Ostrovski, G.; Barreto, A. Temporally-Extended ϵ-Greedy Exploration. In Proceedings of the International Conference on Learning Representations (ICLR), Vienna, Austria, 4–8 May 2021.
40. Zhang, W.; Zhou, D.; Li, L.; Gu, Q. Neural Thompson Sampling. In Proceedings of the International Conference on Learning Representations (ICLR), Vienna, Austria, 4–8 May 2021.
41. Schaul, T.; Quan, J.; Antonoglou, I.; Silver, D. Prioritized Experience Replay. In Proceedings of the International Conference on Learning Representations (ICLR), San Juan, Puerto Rico, 2–4 May 2016.
42. Andrychowicz, M.; Wolski, F.; Ray, A.; Schneider, J.; Fong, R.; Welinder, P.; McGrew, B.; Tobin, J.; Abbeel, P.; Zaremba, W. Hindsight Experience Replay. In Proceedings of the Advances in Neural Information Processing Systems (NIPS), Long Beach, CA, USA, 4–9 December 2017.